Analysis of Network-on-Chip Topologies for Cost-Efficient Chip Multiprocessors

Marta Ortín-Obón · Darío Suárez-Gracia · María Villarroya-Gaudó · Cruz Izu · Víctor Viñals-Yúfera

Abstract As chip multiprocessors accommodate a growing number of cores, they demand interconnection networks that simultaneously provide low latency, high bandwidth, and low power. Our goal is to provide a comprehensive study of the interactions between the interconnection network and the memory hierarchy to enable a better co-design of both components. We explore the implications of the interconnect choice on overall performance by comparing the behaviour of three topologies (mesh, torus, and ring) and their concentrated versions. Simply choosing the concentrated mesh over the ring improves performance by over 40% in a 64-core chip.

The key strength of this work is the holistic analysis of the network-on-chip and the memory hierarchy. Experiments are carried out with a full-system simulator that carefully models the processors (single and multithreaded), memory hierarchy, and interconnection network, and executes realistic parallel and multiprogrammed workloads. We corroborate conclusions from several previous works: network diameter is critical, the concentrated mesh offers the best area-energy-delay trade-off, and traffic is very light and highly unbalanced. We also provide interesting insights about application-specific features that are hidden when studying only average results. We include a fairness analysis for multiprogrammed applications, and refute the idea of the memory controller placement greatly affecting performance.

Keywords: interconnection networks, chip multiprocessor, topology, mesh, torus, ring

1 Introduction

Nowadays, a single chip may contain multiple processors and a significant amount of memory. A popular trend consists of interconnecting several nodes, each of them with a core and one or more levels of private and/or shared cache memories. Nodes communicate through an interconnection network that allows them to exchange coherence messages and cache blocks, and has a major impact on overall performance, energy consumption, and area. We focus on general purpose CMPs, where both high-performance and low-power are required in equal shares.

Only a few works study the interconnect by modelling in detail the processors, memory hierarchy, and interconnection network. However, those analysis are often performed with synthetic traffic or

M. Ortín-Obón (corresponding author) · M. Villarroya-Gaudó · V. Viñals-Yúfera
Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza
Tel.: +34-876555341
E-mail: {ortin.marta, mvg, victor}@unizar.es

D. Suárez-Gracia
Qualcomm Research Silicon Valley
E-mail: dario@unizar.es

C. Izu
Department of Computer Science, University of Adelaide
E-mail: cruz@cs.adelaide.edu.au

application traces that do not entirely capture the behaviour of a real execution [10,25,30,6]. This work simulates both parallel and multiprogrammed workloads with real applications, carefully modelling all the components above-mentioned. This allows us to study the effect of the interconnection network configuration on the whole system and the real interactions between the memory subsystem and the interconnect. We revisit the comparison of several topologies with our detailed simulation framework to update the results, validate or refute previous conclusions, and complete them with further analysis. We present an analysis of three topologies with varying degrees of complexity, performance, power, and area: mesh, torus, and ring. We model CMPs with 16 and 64 single-threaded cores, including a configuration with 16 4-threaded cores, and explore the effect of modifying the location and number of memory controllers. Our goal is to draw meaningful conclusions on the studied network configurations and study the details, pointing out the best choice from an integrated performance, area, and energy standpoint.

The rest of this document is organized as follows: Section 2 presents the related work; Section 3 describes the CMP architecture and the interconnection network configuration; Section 4 introduces the methodology followed in this work; Section 5 describes the qualitative analysis of the topologies; Section 6 explains our simulation results, and Section 7 concludes the paper.

2 Related work

Several publications have highlighted the impact of the network on performance, energy, and chip area. However, only a few papers focus on the comparison of interconnection network configurations. Balfour and Dally present an analysis of how different topologies affect performance, area, and energy efficiency [6]. However, they do not model the memory subsystem, only use synthetic traffic patterns, and do not consider simple topologies like the ring. Gilabert *et al.* focus on physical synthesis of several networks, but do not simulate real applications or systems larger than 16 cores [16]. Villanueva *et al.* highlight the importance of a comprehensive simulation framework and present results of the execution of real parallel applications and its close relationship with cache behaviour [41]. Sanchez *et al.* explore the implications of interconnection network design for CMPs [36]. We complement their results including a simple topology (ring), multiprogrammed workloads, traffic distribution analysis, the effect of memory controller placement, and the influence of the network topology on fairness.

Many papers propose alternatives to conventional router architectures, topologies, and flow control methods on isolation. However, they do not consider the impact on the overall system and back up the results with network-only simulations of synthetic traffic and traces. Carara *et al.* revisit circuit-switching which, as opposed to packet-switching, allows to reduce buffer size, and guarantees throughput and latency [10]; Walter *et al.* try to avoid hotspots on systems on chip by implementing a distributed access regulation technique that fairly allocates resources for certain modules [42]; Mishra *et al.* propose an heterogeneous on-chip interconnect that allocates more resources for routers suffering higher traffic but they only get good results with a mesh topology [33]; Koibuchi *et al.* detect that adding random links to a ring topology results in big performance gains, although they only experiment with a network simulator [25]. All these studies either do not model the whole system, do not include a significant variety of real workloads, or do not experiment with different topologies. Also, most of them only include network-related metrics and fail to report on overall performance, or elaborate conclusions based on IPC (instructions per cycle), which has been reported to be unsuitable for parallel applications [47].

Another approach consists on designing the network considering the behaviour of the memory subsystem and the coherence protocol. Yoon *et al.* propose an architecture with parallel physical networks with narrower links and smaller routers that eliminates virtual channels [45]. Seiculescu *et al.* propose to use two dedicated networks: one for requests and one for replies [37]. Lodde *et al.* introduce a smaller network for invalidation messages, but only test their design with memory access traces [30]. Agarwal *et al.* propose embedding small in-network coherence filters inside on-chip routers to dynamically track sharing patterns and eliminate broadcast messages [5]. These studies try to improve the performance of the most commonly used networks, but do not venture with less conventional topologies. Also, they only experiment with a maximum of 16 cores. Krishna *et al.* propose a system

to improve the frequent 1-to-many and many-to-1 communication patterns by forking and aggregating packets to avoid the increment in traffic as the number of nodes increases [26]. Bezerra *et al.* try to reduce traffic by statically mapping memory blocks to physical locations on the chip that are close to cores that access them [8]. The last two proposals are only evaluated with a typical mesh topology.

3 CMP Architecture Framework

This section presents the modelled CMP architecture and a detailed description of all the interconnection network configurations.

3.1 General System Architecture

Our study focuses on homogeneous CMPs. The system is composed of several tiles connected by an interconnection network. Each tile has a core with a private first level cache (L1) split into data and instructions and a bank of the shared second level cache (L2), both connected to the router. In the initial setting, four tiles in the edges of the chip also include a memory controller. Figure 1 depicts the block diagram of the chip and a tile with memory controller. It also includes the connections between the elements in the tile and the router. Table 1 summarizes the key parameters of the system. To model the architecture we based our design on other systems with similar characteristics, both from academia [46,37,7] and industry (Tilera's TILE*Pro*64 [40], Intel Xeon Phi [20], and Intel 48-core processor [19]). To size our L2 cache (which is our last level cache) we have taken a configuration very frequently used in academia [1,2,22] that is also a nice compromise among the sizes of shared last level caches in high and low-end commercial platforms. For example, the AMD Opteron processor has a shared L3 cache of 6 MB for 6 cores [11]; IBM Power8 has 8 to 12 cores with 8 threads per core, and includes an L3 cache with 64 to 96 MB, as well as an L4 cache with 32 to 64 MB [18]; Intel Xeon D has 1.5 MB of L2 cache per core [24]; Sparc M7 has 32 cores and 64 MB of shared L3 cache [34].

An interesting design trend for CMPs is to integrate a large number of nodes by using simple cores. That is why we are modelling systems with 16 and 64 Ultrasparc III Plus single-thread in-order cores. However, we also consider the effect of multithreading by simulating a configuration with 16 cores with 4 threads each.

We use a directory-based MESI coherence protocol. All the traffic that traverses the interconnection network is a direct consequence of the memory activity, either to move cache lines (instructions or data) among tiles or for coherence management. Therefore, it is important to model the caches realistically, even though our main interest lies in the interconnect [28,36].

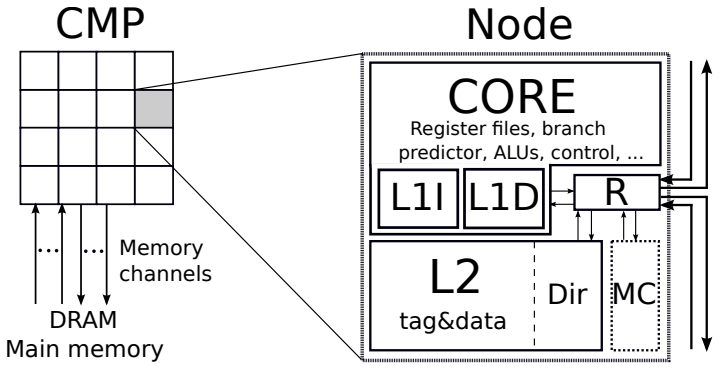


Fig. 1: Block diagram including a chip and the components of a tile. MC stands for memory controller, R is the router, and Dir is the directory, which is included in the L2 cache. This example router has two input and two output ports connected to other neighboring tiles.

Table 1: Main characteristics of the CMP.

Cores	16 single and multithreaded cores, and 64 single-threaded cores, Ultrasparc III Plus, in order, 1 instruction/cycle and thread, 2GHz frequency
Coherence protocol	Directory-based, MESI, directory distributed among L2 cache banks
Consistency model	Sequential
Private L1 cache	32KB data and instruction caches, 4-way set associative, 2-cycle hit access time, 64B line size, pseudo-LRU replacement policy
Shared L2 cache	Physically distributed, 1 bank/tile, 1MB per bank, 16-way set associative, 64B line size Pseudo-LRU replacement policy, inclusive, interleaved by line address 7-cycle hit access time
Memory	4 memory controllers, distributed in the edges of the chip, (both for 16 and 64-core architectures), 160-cycle latency Section 6.7 considers different number and location of memory controllers

Table 2: Main characteristics of the interconnection network.

General	Two virtual networks (requests and replies), 2 virtual channels (VCs) per virtual network
Routers	4-stage pipeline: routing and input buffering, VC alloc, switch alloc, and switch traversal Round-robin 2-phase VC/switch allocators 5-flit buffers per VC, they store an entire message (3-flits per buffer in the ring with higher bandwidth)
Links	16-byte flit size (link width), we also consider a higher bandwidth ring with 24B flit size, 1-cycle latency

3.2 Router Architecture

Since our focus is in network topology, we use the same balanced pipelined router in all configurations, except for some exceptions described later, that minimizes the effects of routing in performance. Flits go through four stages in the router: input buffering and routing, virtual channel (VC) allocation, switch allocation, and switch traversal [13] and move following X-Y routing with *wormhole credit-based* flow control. The network runs at 2 GHz, using the same clock frequency as the processors. Table 2 sums up the main parameters of our interconnection network and routers. We use two separate virtual networks to separate traffic classes in order to avoid protocol deadlock. In practice, this means that we will need at least as many virtual channels as virtual networks. In our case, we include 2 virtual channels per virtual network (a total of 4 virtual channels) to improve performance by reducing head-of-line blocking.

4 Methodology

This section describes the simulation environment and workloads used in this work.

4.1 Simulation Environment

We use Simics to perform full-system simulation with 16 (single and multi-threaded, the latter with 4 threads) and 64 (single-threaded) cores [31]. We include GEMS to model the memory subsystem [32], and GARNET for the interconnection network [4]. To get the timing, area, and energy expended by the network we use DSENT [39], a state-of-the-art circuit modelling tool (with 32 nm technology).

Table 3: Simulated workloads and execution methodology.

	Description	16 cores 1-thread	16 cores 4-threads	64 cores 1-thread
Parallel Workloads	From PARSEC: `blackscholes`, `bodytrack`, `canneal`, `dedup`, `ferret`, `fluidanimate`, `raytrace`, `swaptions`, `vips`, and `x264` From SPLASH2: `barnes`, `cholesky`, `fft`, `lu_cb`, `lu_ncb`, `ocean_cp`, `ocean_ncp`, `radiosity`, `radix`, `raytrace`, `volrend`, `water_nsquared`, and `water_spatial` Threads are automatically mapped to the cores by the operating system Simulate the whole parallel region	16 threads	64 threads	64 threads
Multiprog. Workloads	From SPEC CPU2006: `perlbench`, `bzip2`, `gcc`, `mcf`, `sjeng`, `libquantum`, `bwaves`, `milc`, `zeusmp`, `leslie3d`, `dealII`, `soplex`, `GemsFDTD`, `lbm`, `wrf`, and `sphinx3` 20 different mixes with the applications randomly distributed among the cores Applications are bound to the cores to avoid migration Caches are warmed up for 200 million cycles and then, applications are executed for 500 million cycles	16 apps.	16 apps, 4 times each (64 total)	16 apps, 4 times each (64 total)

4.2 Workloads

CMPs can execute parallel applications to reduce execution time, and multiprogrammed workloads (execution of independent programs on each core) to increase throughput. PARSEC is a benchmark suite composed of shared-memory parallel applications that focuses on emerging workloads and was designed to be representative of next-generation programs for chip-multiprocessors [9]. SPLASH2 is a mature benchmark suite that contains a variety of shared-memory, parallel, high performance computing, and graphics applications [44]. We use a selection of benchmarks from PARSEC and SPLASH2 with scaled inputs from PARSEC 3.0.

We have used SPEC CPU2006 to build multiprogrammed workloads in which each core runs a different application, so the only network traffic will come from cache misses and replacements [38]. We choose 16 applications with large working sets (according to [17]) to find potential bottlenecks in the interconnect.

Table 3 describes the workloads and their execution methodology for the different configurations under test. Table 4 shows the characterization of the workloads with respect to their behaviour in the memory subsystem. This helps us understand the amount of network traffic the applications generate. The most noticeable aspect is that the multiprogrammed workloads have a much lower L2 hit rate and need to access main memory more often.

Table 4: Characterization of the workloads with respect to their behaviour in the memory subsystem

	Parallel Applications			Multiprogrammed Workloads		
	16 cores 1-thread	16 cores 4-threads	64 cores 1-thread	16 cores 1-thread	16 cores 4-threads	64 cores 1-thread
LD/ST instructions	29.7%	26.3%	26.6%	28.3%	26.9%	26.3%
L1D hit rate	93.1%	92.9%	89.3%	95.4%	94.5%	93.0%
misses served by L2	91.8%	96.8%	95.5%	55.9%	62.1%	47.3%
misses served by main memory	8.2%	3.9%	4.5%	44.1%	37.9%	52.7%

5 Topologies for homogeneous CMPs: Qualitative Analysis

We compare today's most mainstream topologies: mesh, torus, and ring. The *2D mesh* is a widespread choice for large-scale CMPs due to its regularity. Tiles are organized in a regular grid with links pointing to all 4 cardinal directions: north, south, east, and west. A *torus* is a mesh with wraparound links to reduce the average number of hops between tiles, at the cost of longer links ($\sqrt{2}$ times larger than a mesh[13]), larger area, and high power consumption. While often longer links involve higher wiring latency [43], we kept constant the link latency for all topologies after verifying feasibility with DSENT [39].

Driven by the observed low network occupancy and commercial NoCs [20], instead of moving towards higher-performance topologies, we opt for more efficient options to fit the power budget, such as *bidirectional rings*, which require a smaller area but have a larger diameter. Each node has two links, one in each direction of the ring, as represented in Figure 1.

The torus and the ring have cycles in their topologies, which can lead to deadlocks. To avoid them, we implement a deadlock avoidance method by setting a dateline in each cycle where messages will be forced to use a specific virtual channel so that cycles are broken [13, 14].

Table 5 summarizes the main characteristics of the three topologies. Note that the comparison encompasses topologies with different bisection bandwidth, so first, we tune each topology to obtain a realistic design point, and then, we explore the trade-offs between complexity (which results in more bandwidth, power, and area) and performance for all configurations.

The number of input and output ports of the router is a direct indicator of the complexity; the higher the number of ports, the higher the area and expended energy. If we divided the network in two equal parts, the bandwidth we would have between the two parts is what we call the *bisection bandwidth*. A lower bisection bandwidth indicates that communications in the network will be slower. A *hop* in the network is a link the message traverses when going from source to destination. When counting the total number of hops we also include the local links going from the cache to the router, and from the router to the cache, so the minimum hop count is two (which corresponds to the communication between an L1 and an L2 in the same node through the router of that node: first hop from source cache to router and second hop from router to destination cache). The number of hops gives us an idea of the time it will take a message to traverse the network. In the table, we distinguish the maximum distance (also called *diameter*) and the average distance. Besides, the length of the link will have an impact on the power consumed by the network, which is modelled in DSENT.

The simplicity of the ring topology allows us to test two improved designs. As opposed to the mesh and torus, which have 4 input and 4 output ports to the outside of the tile, the ring has only two of each. The number of ports has a direct effect on the amount of buffer space and the complexity of the switch allocator and crossbar. To make use of this idle space, we test a configuration in which we increase the link bandwidth keeping the router area slightly under that of the torus. This results in flits of 24 bytes, which will reduce the number of flits needed per message and, therefore, serialization latency (RING_FLIT24B). Following the same idea, we also include a ring configuration with reduced latency, where we merge the switch allocation and switch traversal stages, resulting in a 3-cycle router (RING_3CYCLE_R).

Connecting several tiles to the same router to build *concentrated* topologies is a popular choice to reduce the network diameter. This choice has been adopted by the new generation of the Intel Xeon Phi [21]. These designs reduce the amount of resources of the network but might introduce contention. We include concentrated versions of the topologies with a concentration factor of 4. To avoid increasing the router radix, we use *external* concentration with local routers, which allows us to maintain routers with a small area and high frequency with only a small performance degradation [27]. For the 16-core chips we implement a concentrated mesh (CMESH), as depicted in Figure 2. Memory controllers are connected directly to the global router. With only four global routers, the concentrated ring topology is equivalent to the CMESH; the concentrated torus would have additional links, but we omit the results because the higher bandwidth does not benefit performance and increases power and area. For 64 cores, we model the CMESH, CTORUS, and CRING. Tables 5 and 6 include the characteristics of the concentrated versions of the topologies.

Table 5: Qualitative comparison of the three topologies for a CMP system with N tiles (we assume that N will always be a perfect square). We include the basic and the concentrated versions of the topologies, with a concentration factor of c. The number of inputs/outputs does not consider tiles with a memory controller, where routers would have one more input and output, or the tiles in the edges of the mesh, where some ports would be left unused. For the concentrated topologies, indicated ports are for the global routers; local routers always have 6 ports. W is the link bandwidth and L is the link length. For the concentrated topologies, we indicate the length of the links that connect the global routers. Note that the local links have been considered in the hop count formulas. Therefore, to go from node 0 to node 1 we need 3 hops: one from cache 0 to router 0, one from router 0 to router 1, and one from router 1 to cache 1. In the average hop count, the +2 in the formulas corresponds to those local links. For the concentrated topologies, the average hop count is detailed in Table 6 due to its complexity.

Topology	Inputs/ outputs	Bisection BW	Max. hops (diameter)	Avg. hops (Avg distance)	Link length
2D mesh	6/6	$2W\sqrt{N}$	$2\sqrt{N}$	$\sim 2/3\sqrt{N}+2$	L
Torus	6/6	$8W\sqrt{N}$	$\sqrt{N}+2$	$\sim 1/2\sqrt{N}+2$	$L\sqrt{2}$
Ring	4/4	$4W$	$N/2+2$	$\sim N/4+2$	L
CMESH	6/6	$2W\sqrt{N/c}$	$2\sqrt{N/c}+2$	–	$2L$
CTORUS	6/6	$8W\sqrt{N/c}$	$\sqrt{N/c}+2+2$	–	$2L\sqrt{2}$
CRING	4/4	$4W$	$(N/c)/2+2+2$	–	$2L$

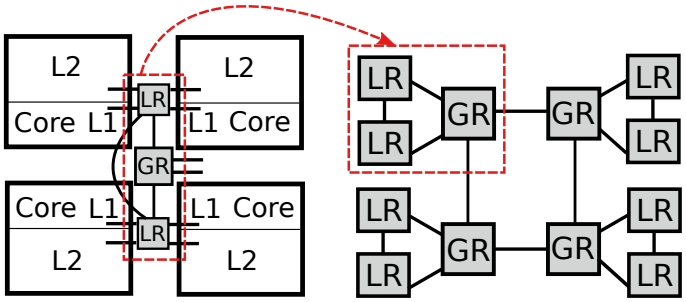


Fig. 2: Connection of the nodes to the routers within a four-node cluster (left) and organization of all local and global routers (right) for a concentrated mesh in a 16-core chip. LR and GR stand for local router and global router, respectively. The lines that connect routers represent always two links, one in each direction.

6 Topologies for Homogeneous CMPs: Quantitative Analysis

This section presents the main contributions of our analysis for 16 and 64-core architectures. We include both system-oriented metrics (performance, area, energy, and fairness) and network metrics (hop count, network latency, and traffic distribution). We conclude with an analysis of the impact of memory controller placement.

6.1 Performance

To compare the impact of the network configurations on performance, we analyse the number of processor cycles it takes for the parallel workloads to complete the parallel section; for the multiprogrammed

Table 6: Average hop count for the concentrated mesh, torus and ring topologies. N is the number of tiles and c is the concentration factor. The formula is divided into the inter and intracluster communications, indicating the probability of each one and the hops in the global network and inside the clusters. Note that the hops to access and leave the network are now 4 (compared with 2 for the non-concentrated topologies), because messages need to traverse the local routers.

	Intercluster Communications			**Intracluster Communications**		
Topology	**Probability**	**Hops global network**	**Hops in cluster**	**Probability**	**Hops global network**	**Hops in cluster**
CMESH	$1-(c/n)$	$2/3\sqrt{N/c}$	4	c/n	0	2.5
CTORUS	$1-(c/n)$	$1/2\sqrt{N/c}$	4	c/n	0	2.5
CRING	$1-(c/n)$	$(N/c)/4$	4	c/n	0	2.5

workloads, we check how many instructions get executed in 500 million cycles. Figure 3 represents the average execution time for the parallel applications and the average CPI (cycles per instruction) for the multiprogrammed workloads, both normalized to the mesh topology. In 16-core single-threaded architectures differences between topologies are small, with the ring with 3-cycle routers and the CMESH being very similar to the mesh, and the torus performing only slightly better. Differences are more pronounced in the multithreaded configurations because the network needs to support a higher load, and topologies with fewer resources are more congested. In 64-core chips, the performance of the ring topologies drops significantly while the concentrated topologies stay very close to the mesh and torus. The conclusions are the same for both parallel and multiprogrammed workloads, and they are along the same line as the most recent industry developments: for the second generation of the Xeon Phi multicore processor, Intel has replaced the ring with a concentrated 2D mesh [20,21].

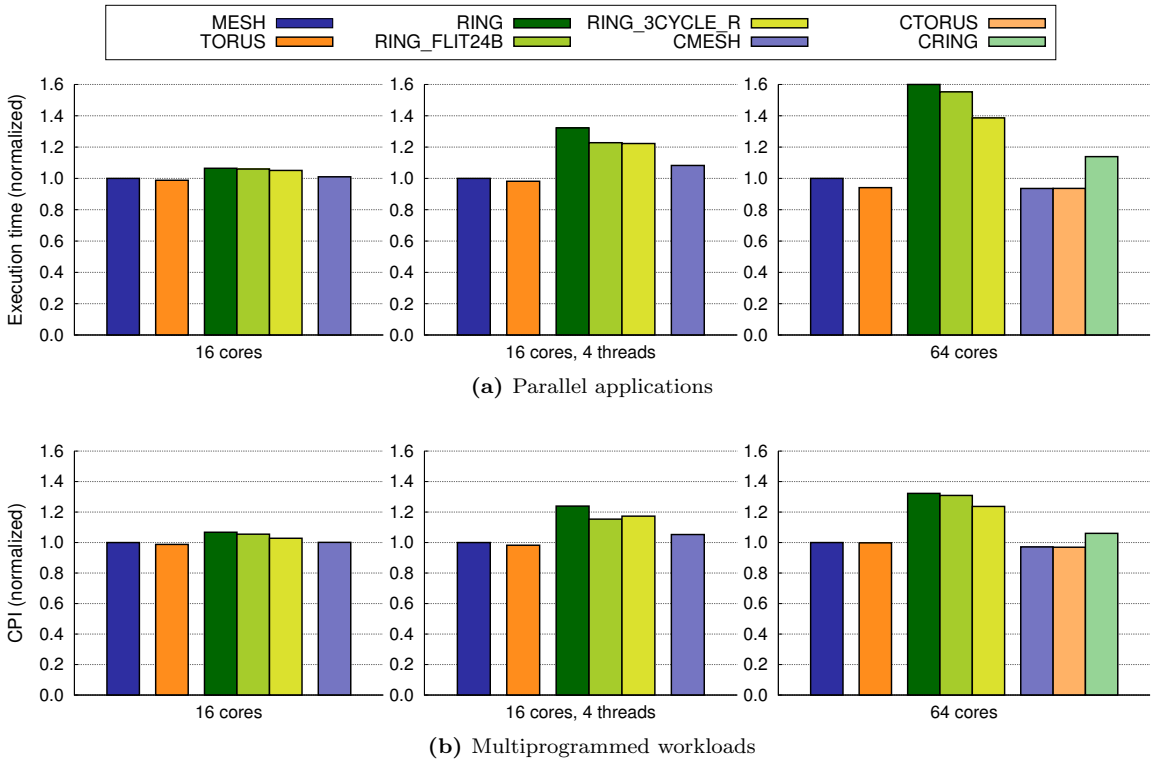
Regarding the absolute CPI values for the multiprogrammed workloads, the CPI of each core in the mesh topology is 4.4 for 16 cores with 1 thread, 4.7 for 16 cores with 4 threads, and 6.1 for 64 cores. Even though the miss rate is small, the penalty of a cache miss greatly increases the CPI, with the highest portion of the miss latency coming from the network latency. Therefore, we can conclude that the impact of the NoC on performance is large.

In the following sections we analyse network-specific metrics and demonstrate how they influence the system performance. We show that the network is lightly loaded and that performance is a direct consequence of the number of hops it takes a message to go from source to destination.

6.2 Average Hop Count

The diameter of the network is critical and concentrated topologies offer faster communications even though messages have to share a smaller amount of routers and links. In Figure 4 we present candlestick charts for the hop count of all the configurations, which show the minimum and maximum values, and the three quartiles. The differences among the workloads are very small. The first thing we notice is that the hop count directly reflects the performance results, showing that this metric determines the performance. Both the median and the variability of the hop count are much larger for the ring topologies, especially with 64 cores, which is a clear indicator of where we experience more pronounced performance drops. Hop count for the three ring topologies is the same in all cases because the variations do not affect the topology. However, performance is different because for the ring with 24-byte flits, data messages are only three flits long instead of five, which reduces the serialization latency; for the ring with 3-cycle routers, every hop takes a smaller number of cycles.

The high impact of the hop count and, more generally, the network latency, is partly due to the simple in-order cores of our system. More complex cores capable of running instructions out-of-order and non-blocking caches would be able to hide some of the network latency by executing other instructions in parallel with the L2 or memory access. However, the pressure on the caches would

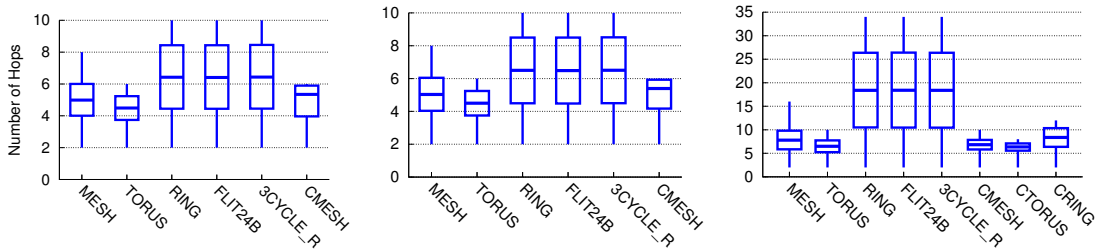


(a) Parallel applications

(b) Multiprogrammed workloads

Fig. 3: Average execution time for the parallel applications and CPI (cycles per instruction) for the multiprogrammed workloads, normalized to the mesh, for 16 single and multithreaded cores and 64 single-threaded cores. In every case, the lower the bars, the better.

not increase enough to give relevance to network throughput, because it has been demonstrated that supporting only 2 in-flight misses is enough to eliminate most of the memory stall cycles [23].



(a) 16 cores **(b)** 16 cores, 4 threads **(c)** 64 cores

Fig. 4: Average hop count for 16 single and multithreaded cores and 64 single-threaded cores. There are no differences between parallel and multiprogrammed workloads, so results are averaged together in all configurations. We present candlesticks, where we can see the minimum, maximum, median, lower quartile (Q1), and upper quartile (Q3) values. Note that scales are different. Hops to traverse the local links from and to the nodes are included in the count.

6.3 Network Latency

Network congestion can delay messages and have a large impact on system performance. Figure 5 represents the *network latency* split in *base latency* (cycles it would take packets to traverse the network without contention), *blocking latency* (extra time spent in the network due to contention), and *queueing latency* (time a message is waiting in the network interface before it can get a free virtual channel to enter the network). The latency for parallel and multiprogrammed workloads is very similar, with the multiprogrammed workloads having slightly higher blocking latency in some cases. That is because these workloads access main memory more often (as we will demonstrate in section 6.4), which creates a bottleneck in the nodes of the network close to the memory controllers.

We can clearly see that the blocking latency is a small percentage of the total latency in the single-threaded configurations: 14% for 16 cores and 16% for 64 cores. It significantly increases with multithreaded cores, reaching an average of 31%, because the higher traffic load creates some congestion, especially in networks with fewer resources (ring and concentrated mesh).

If we focus on the two optimized ring versions, we notice that one of them does not consistently have shorter latency than the other. The 3-cycle router version is normally better, with a shorter base latency. This is because all messages benefit from faster transmissions, while in the 24-byte flit version, only data messages, which need more than one flit, improve their latency. However, in the configurations with 4-threaded cores, the blocking latency is shorter for the ring with 24-byte flits. In those cases when there is more traffic in the network, a nice effect of having larger flits becomes relevant: messages with fewer flits can traverse the network in a more compact way, reducing the number of cycles in which they occupy several routers at a time, thus reducing the probability of conflicts with other messages. This improvement in the blocking latency results in shorter network latency for the 24-byte flit ring in the multithreaded configuration with multiprogrammed workloads.

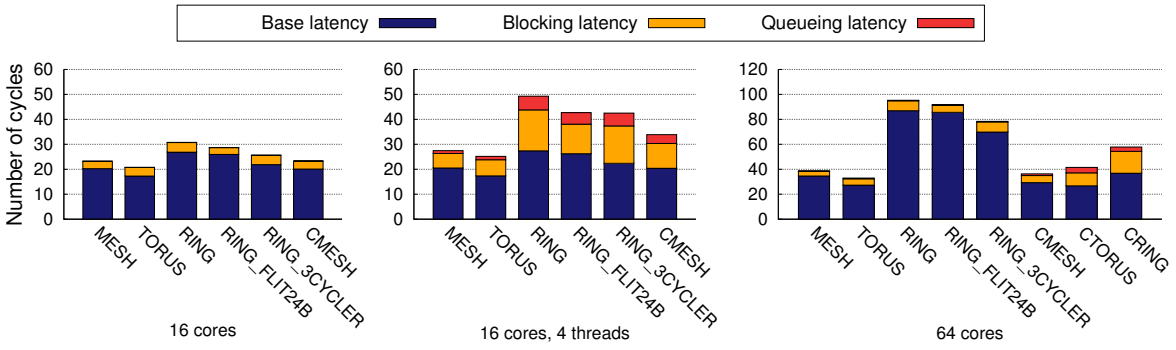
As we already mentioned in the previous section, we can notice again that the network latency results correspond directly with the average hop count and the system performance (the shorter the latency, the better the performance), which demonstrates the huge impact the network has on the system. Nevertheless, the network is mostly lightly loaded, so although it may take long to traverse it, resources are idle most of the time. These results ratify the conclusions of Sanchez *et al.*, which point out that the number of hops is the most critical parameter of the network [36].

6.4 Traffic Distribution

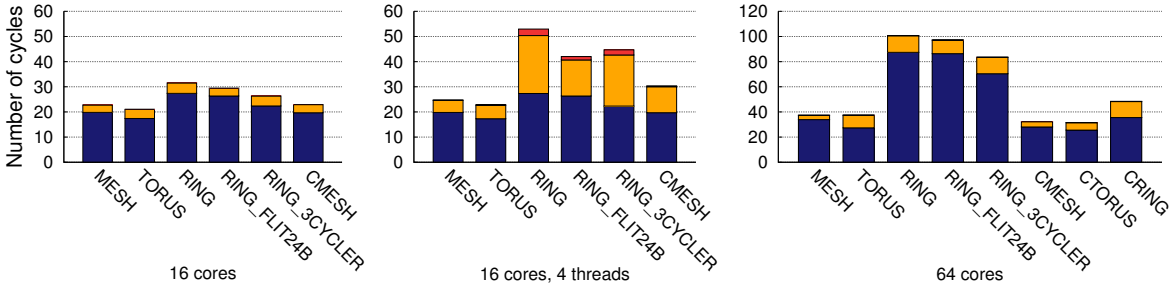
To analyse traffic distribution, we measure the number of injected flits per node. We notice that traffic is unevenly distributed in the interconnect, meaning that some resources will be used more often than others. In this section, we present results for `canneal` as a representative example of parallel applications, and a multiprogrammed mix. For a given application and number of cores, the distribution remains constant when we change the network topology, so we illustrate the results only for the mesh, torus, ring, and CMESH. Conclusions still hold for all applications and number of cores, so we focus on a 64-core chip.

Figure 6 depicts a heat map of injected flits per cycle for each node for `canneal` and a multiprogrammed mix executed on 64 cores. All the traffic is generated by the memory subsystem, so every action has a reaction (request-reply, invalidation-ack). Hence, the heat maps also indicate which nodes are receiving messages more often. The number of flits per cycle is smaller for the ring because a very similar amount of traffic gets injected in a much larger period of time. Nevertheless, the distribution of traffic is the same regardless of the topology: certain nodes inject more flits than others. In the parallel workloads such as `canneal` (Figures 6a to 6d), this is because a couple of L2 banks are being accessed more frequently than others, which depends on the physical distribution of the data touched by each application. The rest of the simulated parallel applications exhibit similar patterns, with 2 out of the 64 nodes injecting more than 40% of the traffic in many applications.

Figures 6e to 6h show results for one of the multiprogrammed mixes. In this case, we see four clear hotspots in the edges of the chip, where the memory controllers are located. The multiprogrammed workloads access main memory more often than parallel applications. Apart from that, the rest of



(a) Parallel applications



(b) Multiprogrammed workloads

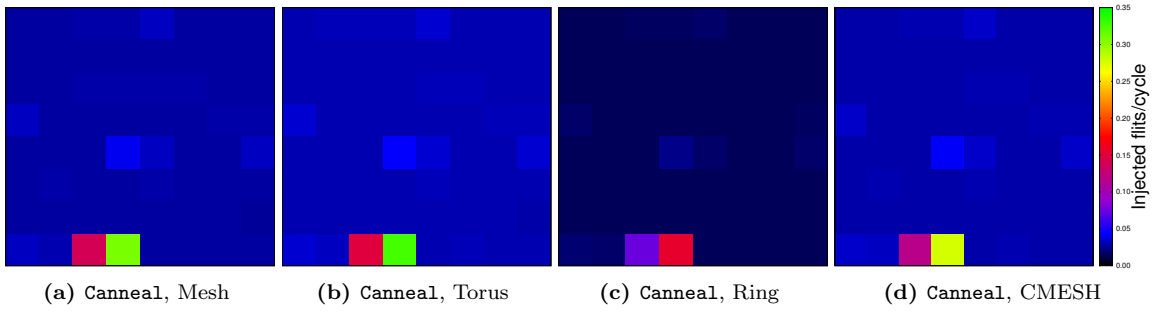
Fig. 5: Average network latency in number of cycles broken down into base, blocking, and queueing latency for 16 single and multithreaded cores and 64 single-threaded cores. Note that scales are different.

ideas we introduced for parallel workloads are still valid. There are several parallel applications which have a larger working set and need to access main memory more often (`fft`, `ocean_cp`, `ocean_ncp`, and `radix`). For those applications, we also see more flits injected from the nodes with memory controllers.

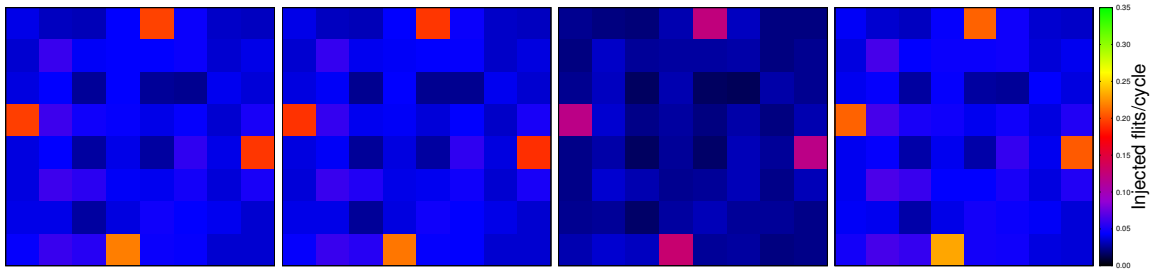
Figure 7 depicts the number of flits per cycle that traverse the network links for `canneal` and multiprogrammed mix. We see that link utilization is higher around the nodes with higher injection rates. Also, it is higher in the ring topologies, since there are fewer links to transport the same amount of information. The torus wastes more resources because it shows the lowest link usage, even though it injects the highest number of flits per cycle. In the execution of the multiprogrammed workload in the mesh topology, we detect that links are used more often in the center of the chip, which is the characteristic behaviour for this topology with uniform traffic. The location of the memory controllers in the edges of the chip increases link usage in the center.

Evaluating all the results of this section, we notice that the network is lightly loaded, even around the most active nodes; furthermore, some parts of the network are idle most of the time. Considering all applications executed on single-threaded architectures, nodes in parallel and multiprogrammed workloads inject an average of 0.024 and 0.052 flits per cycle, respectively; for our 16 multi-threaded core configuration, they inject 0.11 and 0.23 flits per cycle on average. For comparison, Dally shows that the saturation throughput of an 8x8 mesh with four 1-flit virtual channels is around 0.5 and 0.6 [12], and that value would be even lighter with larger VCs. Since the network is lightly loaded, congestion delays are not a major contributor to network latency. Even on the multithreaded configurations where there is more traffic in the interconnect, we still see the same relative performance among the topologies, pointing out the paramount importance of the network diameter. This explains why the concentrated topologies reduce network distance without a significant increment on network contention.

Our results show that real workloads exhibit non-uniform traffic patterns across the network, even though this cannot be perceived when considering only average statistics. Instead of the fairly uniform traffic distributions seen with synthetic networking workloads, we observe hotspots in some locations.

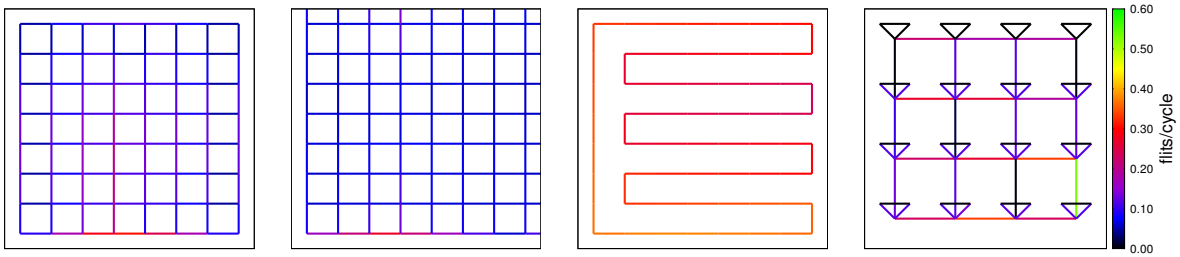


(a) `Canneal`, Mesh **(b)** `Canneal`, Torus **(c)** `Canneal`, Ring **(d)** `Canneal`, CMESH

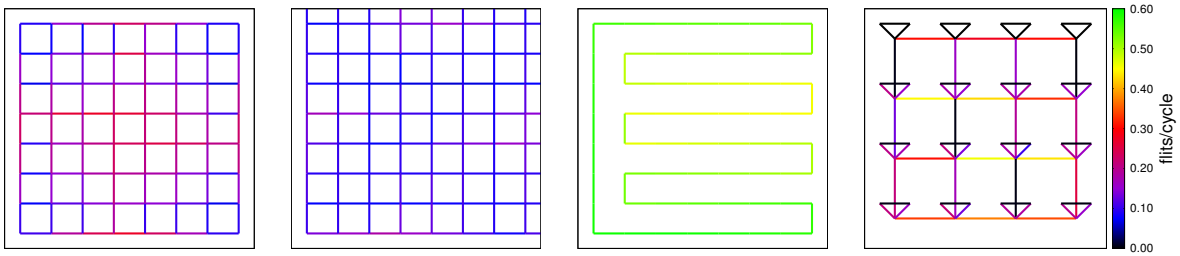


(e) Multi, Mesh **(f)** Multi, Torus **(g)** Multi, Ring **(h)** Multi, CMESH

Fig. 6: Injected flits per cycle and node for the `canneal` parallel application (top) and a multiprogrammed mix (bottom) executed in 64 cores.



(a) `Canneal`, Mesh **(b)** `Canneal`, Torus **(c)** `Canneal`, Ring **(d)** `Canneal`, CMESH



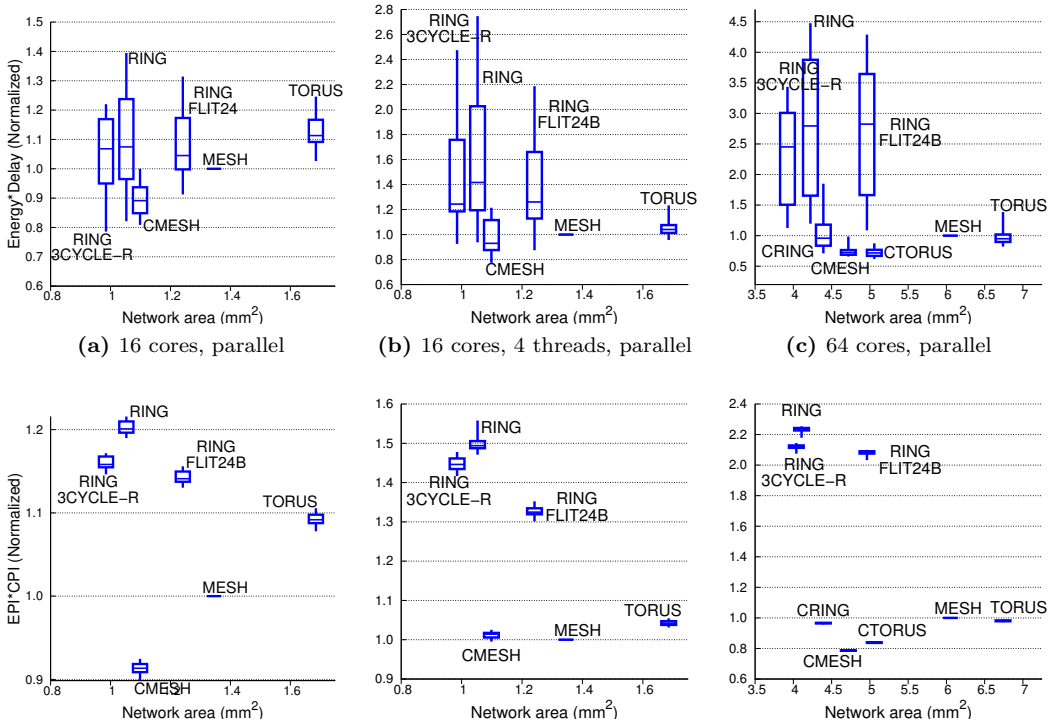
(e) Multi, Mesh **(f)** Multi, Torus **(g)** Multi, Ring **(h)** Multi, CMESH

Fig. 7: Link utilization in flits per cycle for the `canneal` parallel application (top) and the multiprogrammed mix (bottom) executed in 64 cores. Each line is the combination of two links, one in each direction. Injection and ejection links have been left out. In the torus, links that touch the edges of the chip represent the wraparound links. In the CMESH, lines that make up triangles are connections between the local and global routers of each cluster of cores.

This points out that synthetic traffic patterns should have hotspots in both flit injection and destination distribution in order to reflect the real traffic load imposed on the network by parallel and multiprogrammed workloads. They also show the potential of fine-grained network reconfiguration for real applications. For instance in the form of dynamic resource allocation or frequency/voltage scaling (DVFS), where some parts of network save power while others increase execution speed. Recent studies also confirm this potential: Lee *et al.* use DVFS for thermal management in 3D ICs, both in the cores and routers, but they do not consider parallel applications [29]; Haghbayan *et al.* also propose DVFS control to honour power and thermal constraints in a dark silicon context, but exclude the network from such control and consider a synthetic task generation model instead of real workloads [35]. Reconfiguration is beyond the scope of the paper but our data adds experimental evidence to its great interest.

6.5 Area, Energy, and Delay

When making design choices for future architectures we need to consider performance, power, and area. For parallel applications, we calculate Energy$_{Network}$*Delay$_{ParallelSection}$ (ED); for multiprogrammed workloads, where we simulate a constant number of cycles, we use EPI*CPI (EPI=Energy$_{Network}$ per Instruction, CPI=Cycles per Instruction). Figure 8 depicts network area (as reported by DSENT) versus ED or EPI*CPI normalized to the mesh. To display the variance across the parallel applications and the multiprogrammed mixes, we represent the results with candlesticks. Ideally, we would like our configuration to be in the bottom left corner of the graphs.



(a) 16 cores, parallel **(b)** 16 cores, 4 threads, parallel **(c)** 64 cores, parallel

(d) 16 cores, multiprogrammed **(e)** 16 cores, 4 th, multiprogrammed **(f)** 64 cores, multiprogrammed

Fig. 8: Area versus Energy*Delay for the parallel applications (top) and EPI*CPI for the multiprogrammed workload (bottom) for 16 single and multithreaded cores and 64 single-threaded cores, normalized to the mesh. The RING and RING_3CYCLE_R have the same area, but candlesticks have been slightly shifted on the horizontal axis for better visualization, both have an area of $1.0mm^2$ for 16 cores and $4.1mm^2$ for 64 cores.

For 16 single-threaded cores (plots 8a and 8d), the CMESH offers the lowest values for energy and delay, with a small area (only 8% bigger than the ring and 18% and 35% smaller than the mesh and torus, respectively). For 16 multi-threaded cores (plots 8b and 8e) and, especially for 64 cores (plots 8c and 8f), the ED and EPI*CPI increase substantially for the ring topologies with all workloads. The Delay contributor increases much more significantly with more cores due to the higher hop count. Therefore, networks with lower diameter perform better when integrating a larger number of cores. In this case, the CMESH still offers the best trade-offs. We also see that the variance across the multiprogrammed mixes is very small, pointing out that the way of distributing independent applications in the chip does not impact either performance or network energy. Our results show that over-dimensioning the network is not the best solution: a simple topology like the CRING is better than the torus from all standpoints. Even in the multithreaded architecture (plots 8b and 8e), where the network has a higher load, the CMESH still offers a better trade-off than the torus.

We also see that the deviation of the results varies among topologies and is bigger with 64 cores. It is proportional to the variation in network latency, which increases with the average distance of the network and hop latency. This is because the Delay component of the ED product suffers bigger increases in certain applications where the thread distribution generates disadvantageous traffic patterns for the ring topology.

6.6 Fairness

In a multiprogrammed environment, fairness determines if resources are evenly distributed among independent applications. A system is fair if all the multiprogrammed applications experience an equal slowdown compared to their performance when executed alone. Our interest lies in assessing whether the topology influences fairness. To numerically quantify fairness, we rely on the following formula:

$$fairness = \frac{\min\limits_{i}\left(\frac{CPI_i^{MT}}{CPI_i^{ST}}\right)}{\max\limits_{i}\left(\frac{CPI_i^{MT}}{CPI_i^{ST}}\right)}$$

where CPI, ST, and MT refer to cycles per instruction, single thread, and multi-threaded execution, respectively [15]. The i index refers to the applications. The ideal value would be 1; the closer we are to it, the better fairness we have. To calculate the fairness, we take one of the mixes and simulate both the mix and each application running alone in chip, pinned to the same core in both cases.

Figure 9 shows the fairness for all the topologies on 16 single and multithreaded cores, and 64 single-threaded cores according to that formula. In order to evaluate if there are any outlier numbers that are negatively affecting the final results, we present in Figure 10 candlesticks with the ratio between the CPI of the applications executed along with the rest of applications (CPI-MT) and the CPI of the same application running alone in the chip (CPI-ST). In this case, fairer configurations will present short candlesticks, while unfair networks will have big and long candlesticks. The same conclusions can be extracted from both graphs. With lightly loaded networks, fairness is very similar across all topologies. It significantly decreases in two cases: for the multithreaded cores respect to the single-threaded configurations, and for the ring topologies respect to the others with 16 multithreaded cores. These reductions correspond to cases where the network supports a higher load. In those situations, there is more congestion on the network, and that has a higher impact on the applications that need to use it more often, while others with more hits on their first level cache remain undisturbed. In the multithreaded cores, we clearly notice a correlation between performance and fairness: the mesh, torus, and CMESH show the highest fairness because more efficient networks can successfully support more simultaneous communications without interferences.

6.7 Memory Controller Placement

In the previous sections, all configurations had four memory controllers located at the edges of the chip. It has been demonstrated that the location of the memory controllers impacts memory latency in

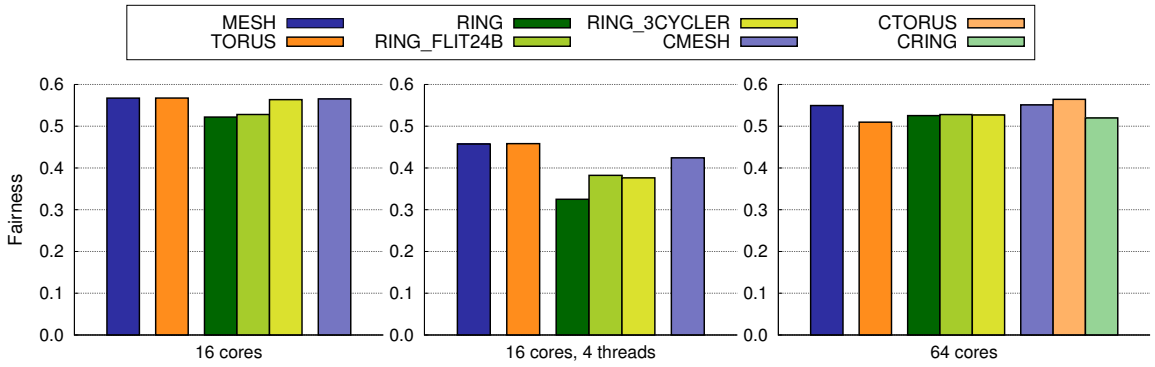
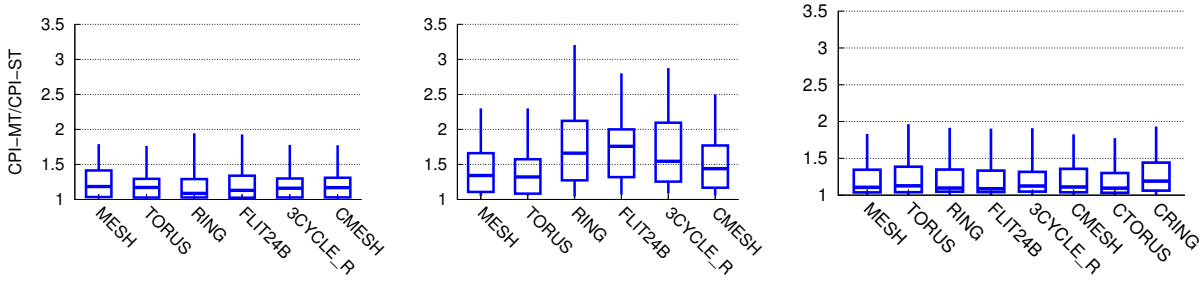


Fig. 9: Fairness for the multiprogrammed workloads in chips with 16 single and multithreaded cores and 64 single-threaded cores. The higher the bars, the better.



(a) 16 cores **(b)** 16 cores, 4 threads **(c)** 64 cores

Fig. 10: Fairness for the multiprogrammed workloads in chips with 16 single and multithreaded cores and 64 single-threaded cores, represented by the ratio of the CPI of the applications executed along with the rest of applications (CPI-MT) divided by the CPI of the same application running alone in the chip (CPI-ST). We present candlesticks, where we can see the minimum, maximum, median, lower quartile (Q1), and upper quartile (Q3).

a mesh topology [3]. We have compared several options varying the number and placement of memory controllers to look for the best configuration in terms of performance. For the sake of brevity, this section focuses on multiprogrammed workloads because they exhibit higher main memory access rate. Also, we limit the design space to the mesh and CMESH topologies, because they are the ones that offer the best performance, energy, and area trade-offs (as we showed in section 6.5), and to the 64-core chip, where distances are longer and MC placement has a larger impact. Figure 11 shows the nodes of the chip where the memory controllers are located for the mesh and the CMESH topologies. We test 9 configurations for the mesh, with 4, 8, and 16 memory controllers; for the CMESH we test 5 configurations with 4, 8, and 16 controllers. In the CMESH, the MCs are connected directly to the global routers (see Section 6), so we divide the chip in only 16 squares, which represent clusters of 4 cores each.

We calculate the average performance of the multiprogrammed workloads with all the memory controller configurations, and see that variations in performance are so small (always smaller than 0.03%) that both the number and placement of memory controllers seem to be a secondary issue. This is because benchmarks do not miss in the L2 very often, even for the multiprogrammed workloads, which is where we detected the largest amount of traffic to memory. Increasing the number of memory controllers does not have a significant impact on performance either.

Abts *et al.* state that memory controller placement is critical and that a good placement reduces contention, lowers network latency, and provides predictable performance [3]. Although they simulate

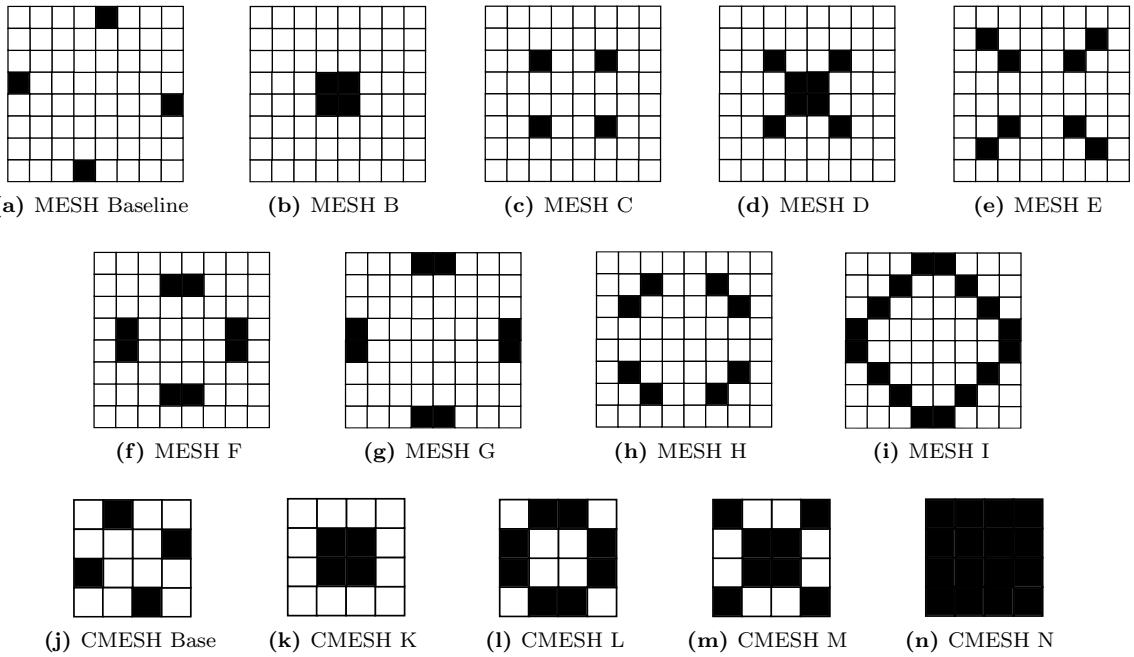


Fig. 11: Memory controller configurations for the mesh -(a) to (i)- and CMESH -(j) to (n)- topologies with 64 cores. The shaded tiles include a memory controller. For the CMESH, MCs are connected to the global routers so we only represent the 4-core clusters of the chip.

different workloads than we do and with smaller caches, we have compared the amount of traffic to memory they have to what we see in our simulations. We have determined that our multiprogrammed workloads access main memory more often than their applications, increasing the effect of memory controller placement on overall performance. It is true that some memory controller placements reduce network latency, but we go a step further and guarantee that the impact on system performance is negligible for such light traffic to memory.

Our main objective was to determine if the memory controller placement is relevant for this kind of general purpose CMPs with these applications, which represent reasonable workloads. However, it is also interesting to determine if the memory controllers could significantly affect performance on a scenario with more traffic to memory. In order to analyse this, we repeat the experiments with different system parameters: reduced L2 cache size to increase the traffic to memory (down to 128 KB per core from the original 1 MB per core), and faster memory access to increase the impact of the network latency reduction (50 cycles instead of the baseline 150). These new simulations increase the traffic to memory by 50%, but the effect of the memory controller placement is still small. In average, the difference in performance with respect to the baseline is 0.6%, which is much more than what we were seeing before, but still very small. Therefore, we conclude that the memory controller placement does not have a relevant impact on performance in our general purpose CMP, even with system parameters that enlarge the impact of the network on memory access time. In any case, our results do not rule out the importance of this issue in specific situations, such as streaming memory applications or systems with even smaller caches.

7 Conclusions

Considering the interconnection network and the cache hierarchy simultaneously helps identify improvement opportunities in the design of CMPs. Both elements have a significant influence on system performance, area, and power consumption. We have modelled in detail the processors, memory hi-

erarchy, and network using full-system simulation and executing both parallel and multiprogrammed workloads. We have performed a qualitative and quantitative analysis of three network topologies: mesh, torus, and ring, including two additional ring configurations (one with more bandwidth and one with 3-cycle routers) and concentrated networks for CMPs with 16 single and multithreaded cores and 64 single-threaded cores.

Our results show that performance is highly affected by the choice of the interconnect, especially in 64-core systems, where the ring performance drops by 72% with respect to the CMESH for parallel workloads. The ring topologies perform worse due to the increased hop count, which translates into higher network latency. In average, compared with the CMESH, the ring suffers from a 34% network latency increase in the single-threaded 16-core chip, 60% in the multi-threaded 16-core chip, and 136% in the 64-core chip. The CMESH topology offers the best performance with low energy consumption (17% less than the torus for 64 cores) and area (30% smaller than the torus for 64 cores) for all workloads considered and both 16 and 64-core chips, even with multithreaded cores, which generate a heavier traffic load.

We have reported that in real applications traffic is very light and not uniformly distributed, pointing out the potential of heterogeneity, either in the form of dynamic resource allocation or frequency/voltage scaling. For parallel applications, both the injection rate and the message destinations are more variable than those we see with synthetic traffic patterns, with only 2 nodes injecting an average of 33% of the traffic in the 64-core chips; for multiprogrammed workloads, traffic is random with hotspots at the memory controllers, which inject 25% of the traffic.

For multiprogrammed workloads, we have concluded that that contention in the network causes fairness to drop, especially for networks with lower performance. For example, the mesh has 26% lower fairnes with 16 single-threaded cores than with 16 multithreaded cores; focusing only on the multithreaded cores, the ring has 28% lower fairness than the mesh. We have also determined that the placement and the number of memory controllers has a negligible effect on system performance with realistic applications, because they have limited memory access.

8 Acknowledgements

This work was supported in part by grants TIN2013-46957-C2-1-P, Consolider NoE TIN2014-52608-REDC (Spanish Gov.), and gaZ: T48 research group (Aragón Gov. and European ESF), and FPU12/02553. We also thank the anonymous reviewers that greatly helped to improve this work.

References

1. Abousamra, A., Jones, A., Melhem, R.: Codesign of NoC and cache organization for reducing access latency in chip multiprocessors. IEEE Transactions on Parallel and Distributed Systems **23**(6), 1038–1046 (2012). DOI 10.1109/TPDS.2011.238
2. Abousamra, A., Melhem, R., Jones, A.: Deja-vu switching for multiplane NoCs. In: Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS), pp. 11 –18 (2012). DOI 10.1109/NOCS.2012.9
3. Abts, D., Enright Jerger, N.D., Kim, J., Gibson, D., Lipasti, M.H.: Achieving predictable performance through better memory controller placement in many-core CMPs. In: Proceedings of the 36th annual international symposium on Computer architecture, ISCA '09, pp. 451–461. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1555754.1555810. URL `http://doi.acm.org/10.1145/1555754.1555810`
4. Agarwal, N., Krishna, T., Peh, L.S., Jha, N.: GARNET: A detailed on-chip network model inside a full-system simulator. In: IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS, pp. 33 –42 (2009). DOI 10.1109/ISPASS.2009.4919636
5. Agarwal, N., Peh, L.S., Jha, N.K.: In-network coherence filtering: snoopy coherence without broadcasts. In: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 42, pp. 232–243. ACM, New York, NY, USA (2009). DOI http://doi.acm.org/10.1145/1669112.1669143. URL `http://doi.acm.org/10.1145/1669112.1669143`
6. Balfour, J., Dally, W.J.: Design tradeoffs for tiled CMP on-chip networks. In: Proceedings of the 20th annual international conference on Supercomputing, ICS '06, pp. 187–198. ACM, New York, NY, USA (2006). DOI 10.1145/1183401.1183430. URL `http://doi.acm.org/10.1145/1183401.1183430`
7. Barroso, L.A., Gharachorloo, K., McNamara, R., Nowatzyk, A., Qadeer, S., Sano, B., Smith, S., Stets, R., Verghese, B.: Piranha: a scalable architecture based on single-chip multiprocessing. In: Proceedings of the 27th annual

international symposium on Computer architecture, ISCA '00, pp. 282–293. ACM, New York, NY, USA (2000). DOI 10.1145/339647.339696. URL http://doi.acm.org/10.1145/339647.339696
8. Bezerra, G.B.P., Forrest, S., Zarkesh-Ha, P.: Reducing energy and increasing performance with traffic optimization in many-core systems. In: Proceedings of the System Level Interconnect Prediction Workshop, SLIP '11, pp. 3:1–3:7. IEEE Press, Piscataway, NJ, USA (2011). URL http://dl.acm.org/citation.cfm?id=2134224.2134229
9. Bienia, C., Kumar, S., Singh, J.P., Li, K.: The PARSEC benchmark suite: characterization and architectural implications. In: Proceedings of the 17th international conference on Parallel architectures and compilation techniques, PACT '08, pp. 72–81. ACM, New York, NY, USA (2008). DOI http://doi.acm.org/10.1145/1454115.1454128. URL http://doi.acm.org/10.1145/1454115.1454128
10. Carara, E., Moraes, F., Calazans, N.: Router architecture for high-performance NoCs. In: Proceedings of the 20th annual conference on Integrated circuits and systems design, SBCCI '07, pp. 111–116. ACM, New York, NY, USA (2007). DOI 10.1145/1284480.1284515. URL http://doi.acm.org/10.1145/1284480.1284515
11. Conway, P., Kalyanasundharam, N., Donley, G., Lepak, K., Hughes, B.: Cache hierarchy and memory subsystem of the AMD opteron processor. Micro, IEEE **30**(2), 16–29 (2010). DOI 10.1109/MM.2010.31
12. Dally, W.: Virtual-channel flow control. In: 17th Annual International Symposium on Computer Architecture, pp. 60–68 (1990). DOI 10.1109/ISCA.1990.134508
13. Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2003)
14. Dally, W.J., Seitz, C.L.: The torus routing chip. Distributed Computing **1**, 187–196 (1986). URL http://authors.library.caltech.edu/26909/. 10.1007/BF01660031
15. Gabor, R., Weiss, S., Mendelson, A.: Fairness and throughput in switch on event multithreading. In: 39th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-39, pp. 149–160 (2006). DOI 10.1109/MICRO.2006.25
16. Gilabert, F., Medardoni, S., Bertozzi, D., Benini, L., Gomez, M., Lopez, P., Duato, J.: Exploring high-dimensional topologies for NoC design through an integrated analysis and synthesis framework. In: Second ACM/IEEE International Symposium on Networks-on-Chip, NoCs, pp. 107–116 (2008). DOI 10.1109/NOCS.2008.4492730
17. Gove, D.: CPU2006 working set size. SIGARCH Comput. Archit. News **35**(1), 90–96 (2007). DOI 10.1145/1241601.1241619. URL http://doi.acm.org/10.1145/1241601.1241619
18. Halfhill, T.R.: Power8 hits the merchant market. Memory bandwidth helps IBM server processor ace big benchmarks. Microprocessor report (2014)
19. Howard, J., Dighe, S., Vangal, S., Ruhl, G., Borkar, N., Jain, S., Erraguntla, V., Konow, M., Riepen, M., Gries, M., Droege, G., Lund-Larsen, T., Steibl, S., Borkar, S., De, V., Van Der Wijngaart, R.: A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling. IEEE Journal of Solid-State Circuits **46**(1), 173 –183 (2011). DOI 10.1109/JSSC.2010.2079450
20. Intel: Intel Xeon Phi (2014). URL http://www.intel.es/content/dam/www/public/us/en/documents/datasheets/xeon-phi-coprocessor-datasheet.pdf. http://www.intel.es/content/dam/www/public/us/en/documents/datasheets/xeon-phi-coprocessor-datasheet.pdf (Last access November 2015)
21. Intel: Intel Xeon Phi, Knights Landing (2014). URL https://software.intel.com/sites/default/files/managed/e9/b5/Knights-Corner-is-your-path-to-Knights-Landing.pdf. https://software.intel.com/sites/default/files/managed/e9/b5/Knights-Corner-is-your-path-to-Knights-Landing.pdf (Last access November 2015)
22. Jerger, N.D.E., Peh, L.S., Lipasti, M.H.: Circuit-switched coherence. In: Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip, NOCS '08, pp. 193–202. IEEE Computer Society, Washington, DC, USA (2008). URL http://dl.acm.org/citation.cfm?id=1397757.1397999
23. Jouppi, S.L.K.C.J.B.B.N.P.: Performance impacts of non-blocking caches in out-of-order processors. Tech. rep., HP Laboratories (2011). URL http://www.hpl.hp.com/techreports/2011/HPL-2011-65.pdf
24. Kanter, D.: 14nm Xeon D secures the data center. Microprocessor report (2015)
25. Koibuchi, M., Matsutani, H., Amano, H., Hsu, D.F., Casanova, H.: A case for random shortcut topologies for HPC interconnects. In: Proceedings of the 39th International Symposium on Computer Architecture, ISCA '12, pp. 177–188. IEEE Press, Piscataway, NJ, USA (2012). URL http://dl.acm.org/citation.cfm?id=2337159.2337179
26. Krishna, T., Peh, L.S., Beckmann, B.M., Reinhardt, S.K.: Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication. In: Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO-44 '11, pp. 71–82. ACM, New York, NY, USA (2011). DOI 10.1145/2155620.2155630. URL http://doi.acm.org/10.1145/2155620.2155630
27. Kumar, P., Pan, Y., Kim, J., Memik, G., Choudhary, A.: Exploring concentration and channel slicing in on-chip network router. In: Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip, NOCS '09, pp. 276–285. IEEE Computer Society, Washington, DC, USA (2009). DOI 10.1109/NOCS.2009.5071477. URL http://dx.doi.org/10.1109/NOCS.2009.5071477
28. Kumar, R., Zyuban, V., Tullsen, D.M.: Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In: Proceedings of the 32nd annual international symposium on Computer Architecture, ISCA '05, pp. 408–419. IEEE Computer Society, Washington, DC, USA (2005). DOI http://dx.doi.org/10.1109/ISCA.2005.34. URL http://dx.doi.org/10.1109/ISCA.2005.34
29. Lee, J., Ahn, J., Choi, K., Kang, K.: THOR: Orchestrated thermal management of cores and networks in 3D many-core architectures. In: 20th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 773–778 (2015). DOI 10.1109/ASPDAC.2015.7059104
30. Lodde, M., Roca, T., Flich, J.: Heterogeneous network design for effective support of invalidation-based coherency protocols. In: Proceedings of the 2012 Interconnection Network Architecture: On-Chip, Multi-Chip Workshop, INA-OCMC '12, pp. 1–4. ACM, New York, NY, USA (2012). DOI 10.1145/2107763.2107764. URL http://doi.acm.org/10.1145/2107763.2107764

31. Magnusson, P., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J., Larsson, F., Moestedt, A., Werner, B.: Simics: A full system simulation platform. Computer **35**(2), 50 –58 (2002). DOI 10.1109/2.982916
32. Martin, M.M.K., Sorin, D.J., Beckmann, B.M., Marty, M.R., Xu, M., Alameldeen, A.R., Moore, K.E., Hill, M.D., Wood, D.A.: Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. SIGARCH Computer Architecture News **33**, 92–99 (2005). DOI http://doi.acm.org/10.1145/1105734.1105747. URL `http://doi.acm.org/10.1145/1105734.1105747`
33. Mishra, A.K., Vijaykrishnan, N., Das, C.R.: A case for heterogeneous on-chip interconnects for CMPs. In: Proceedings of the 38th annual international symposium on Computer architecture, ISCA '11, pp. 389–400. ACM, New York, NY, USA (2011). DOI 10.1145/2000064.2000111. URL `http://doi.acm.org/10.1145/2000064.2000111`
34. Oracle: Sparc M7-16 Server (2015). URL `http://www.oracle.com/us/products/servers-storage/sparc-m7-16-ds-2687045.pdf`. `http://www.oracle.com/us/products/servers-storage/sparc-m7-16-ds-2687045.pdf` (Last access November 2015)
35. Rahmani, A.M., Haghbayan, M.H., Kanduri, A., Weldezion, A.Y., Liljeberg, P., Plosila, J., Jantsch, A., Tenhunen, H.: Dynamic power management for many-core platforms in the dark silicon era: A multi-objective control approach. In: IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 219–224 (2015). DOI 10.1109/ISLPED.2015.7273517
36. Sanchez, D., Michelogiannakis, G., Kozyrakis, C.: An analysis of on-chip interconnection networks for large-scale chip multiprocessors. ACM Transactions on Architecture and Code Optimization **7**(1), 4:1–4:28 (2010). DOI 10.1145/1756065.1736069. URL `http://doi.acm.org/10.1145/1756065.1736069`
37. Seiculescu, C., Volos, S., Khosro Pour, N., Falsafi, B., De Micheli, G.: CCNoC: On-Chip Interconnects for Cache-Coherent Manycore Server Chips. In: Proceedings of the Workshop on Energy-Efficient Design (WEED 2011) (2011)
38. Standard Performance Evaluation Corporation (SPEC): SPEC CPU2006. URL `http://www.spec.org/cpu2006/`. `http://www.spec.org/cpu2006/` (Last access November 2015)
39. Sun, C., Chen, C.H.O., Kurian, G., Wei, L., Miller, J., Agarwal, A., Peh, L.S., Stojanovic, V.: DSENT - a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In: Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, NOCS '12, pp. 201–210. IEEE Computer Society, Washington, DC, USA (2012). DOI 10.1109/NOCS.2012.31. URL `http://dx.doi.org/10.1109/NOCS.2012.31`
40. Tilera: TILEPro64 (2008). URL `http://www.tilera.com/products/processors/TILEPro_Family`. `http://www.tilera.com/products/processors/TILEPro_Family` (Last access November 2015)
41. Villanueva, J., Flich, J., Duato, J., Eberle, H., Gura, N., Olesinski, W.: A performance evaluation of 2D-mesh, ring, and crossbar interconnects for chip multi-processors. In: Network on Chip Architectures, 2009. NoCArc 2009. 2nd International Workshop on, pp. 51–56 (2009)
42. Walter, I., Cidon, I., Ginosar, R., Kolodny, A.: Access regulation to hot-modules in wormhole NoCs. In: Proceedings of the First International Symposium on Networks-on-Chip, NOCS '07, pp. 137–148. IEEE Computer Society, Washington, DC, USA (2007). DOI http://dx.doi.org/10.1109/NOCS.2007.8. URL `http://dx.doi.org/10.1109/NOCS.2007.8`
43. Weste, N., Harris, D.: CMOS VLSI Design: A Circuits and Systems Perspective, 4th edn. Addison-Wesley Publishing Company, USA (2010)
44. Woo, S.C., Ohara, M., Torrie, E., Singh, J.P., Gupta, A.: The SPLASH-2 programs: characterization and methodological considerations. In: Proceedings of the 22nd annual international symposium on Computer architecture, ISCA '95, pp. 24–36. ACM, New York, NY, USA (1995). DOI http://doi.acm.org/10.1145/223982.223990. URL `http://doi.acm.org/10.1145/223982.223990`
45. Yoon, Y.J., Concer, N., Petracca, M., Carloni, L.: Virtual channels vs. multiple physical networks: a comparative analysis. In: Proceedings of the 47th Design Automation Conference, DAC '10, pp. 162–165. ACM, New York, NY, USA (2010). DOI 10.1145/1837274.1837315. URL `http://doi.acm.org/10.1145/1837274.1837315`
46. Zhang, M., Asanovic, K.: Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors. In: Proceedings of the 32nd annual international symposium on Computer Architecture, ISCA '05, pp. 336–345. IEEE Computer Society, Washington, DC, USA (2005). DOI 10.1109/ISCA.2005.53. URL `http://dx.doi.org/10.1109/ISCA.2005.53`
47. Zhou, P., Yin, J., Zhai, A., Sapatnekar, S.S.: NoC frequency scaling with flexible-pipeline routers. In: Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design, ISLPED '11, pp. 403–408. IEEE Press, Piscataway, NJ, USA (2011). URL `http://dl.acm.org/citation.cfm?id=2016802.2016897`